IV.   Remarks

The specification was objected to as failing to properly identify "JAVA" as a trademark. Claims 1-11 were rejected under 35 U.S.C. §102(a) as being anticipated by Gudmundson. Reconsideration is respectfully requested.

Applicant appreciates the courtesy shown by the Examiner in the telephone interview with applicant's attorney on January 14, 2005. In that interview, the nature of the present invention and the differences between the invention and the cited prior art were discussed. These differences are set forth below.

Applicant respectfully requests withdrawal of the objection to the specification. Applicant has not used the term "JAVA" alone in the sense of the trademark. The term has been used as "Java Messaging Service" or "JSM", which is a known API standard in the industry. The specification uses the term in the same manner as Sun Microsystems, the owner of the JAVA trademark, uses it on its website and other materials. Accordingly, the application does not use a mark in a manner which would adversely affect Sun Microsystem's trademark.

Claims 1-11 were rejected as being anticipated by Gudmundson. Applicant respectfully asserts that the claims patentably distinguish over the cited art.

While both Gudmundson and the present invention relate to systems for developing software, they do it using completely different processes. Gudmundson relates to an object oriented system allowing hierarchical encapsulation of instantiated objects. The hierarchical encapsulation includes Elements, Modifiers, and an order for execution of the Modifiers. The Modifiers provide characteristics and behaviors to the Elements. In order to do so, specific Modifiers are incorporated into a specific instantiation of an Element. Thus, the Modifier becomes an integral part of that instantiated Element. The Modifiers within an Element may be ordered to provide a control hierarchy. Gudmundson further discloses a complex hierarchical structure for Elements and Modifiers, including parent and child object containers for each. Changes to a class of an Element or Modifier propagate through the complex structure so that all appropriate changes are made. Messages are used to pass information to all of the instantiated objects with the hierarchical structure during use of the objects.

The present invention, on the other hand, does not relate to a hierarchy, encapsulation of instantiated objects, or even object oriented programming. Although the present application refers to "objects", it does not use the term in the manner of object oriented technology. It uses the term to represent discrete parts or modules within the programming environment. The present invention relates to a method and system for creating software applications. The way in which software applications are created according to the present invention is unique. Typical systems for creating software, including object oriented programming processes, are data dependent. They require an initial determination of the data structures of the desired application before the functionality of the application is created. The functionality of the application is created through the programming process using the predefined data structure or format. On the other hand, the system and method of present invention are independent of the data structures. A basic concept of the invention is to completely separate the data structure (the object models of basic data types) and the functional models (the service objects) for any software implementation process. This results from the realization that complex data structures can be represented as groupings or sets of a finite number of basic object types. The data for an application can take any format and an infinite number of formats may be created. However, all formats can be represented by the groupings of the basic object types.

Similarly, functional modules, called service objects in the present application, provide the functionality of the application independent of the data structures of the software application. The service objects are created to operate on each of the basic object types. Since service objects operation with respect to each of the basic object types, they become independent of the data structures. Any data provided as an input to a service object can be operated upon. Furthermore, the service objects determine the basic object types of the data from the object models in performing the processing. Since the service objects are independent of the data structures, they reused in any application, without regard to the specific data structures of the application. Complex data structures can easily be accommodated with the programming process of the present invention. For example, the specification, and some of the claims, identify "class" and "object array" as two possible basic object types. These types can be recursive in nature or can vary as to their attributes. Nevertheless, the variations are irrelevant in operation of the service objects. The service objects process the data simply as the basic object

6

types represent by the object mode, without regard to the complexity of the structure itself. The approaches above, in theory, make possible the use and reuse a finite set of service objects to serve the major implementation effort of any software application development with infinite data structures. Furthermore, the finite nature of both basic object types and service objects allows software implementation blocks to move to pattern oriented deployable environments, such as hardware chips.

With the present invention, the creation of application software is greatly simplified and corresponds to three steps as set forth in claim 1. In the first step, the data structures of the application being developed are defined as object models. The object models are sets of elements wherein each element is one of a basic object type. This system allows the object models to provide complex data structures through combinations of a finite set of basic object types. In the second step, a subset of service objects are selected from the previously created service objects for the application. As noted above, the service objects provide the functionality of the application by performing a function with respect to each of the basic object types. Furthermore, the service objects parse a object model to determine the basic object types in that data model and to perform the necessary function for each of those basic object types. The third step of the application software creation process is selecting a flow process which represents an order of operation of the selected service objects. The object models, selected service objects, and flow process define the entire application. When the application executes (not part of the claims), the flow process is merely followed. Each service object is performed in turn according to the define flow process. When a service object is performed, it parses the object model of the data of the application. It performs its function on each of the basic object types within the object model.

Gudmundson does not disclose, teach or suggest the present claimed invention as recited in claim 1. As noted by the Examiner during the telephone interview, the Modifiers in Gudmundson add functionality or behaviors to the basic elements. However, the service objects in the present invention do not add functionality to the data. Rather, they provide the functionality of the application. In Gudmundson, Modifiers are associated with Elements to create characters or scenes. The Elements and Modifiers are further associated with other

Elements and Modifiers in a hierarchical structure. The service objects of the present invention are not associated with the data to provide specific functionality to that data. Rather, the service objects operate in the order defined by the flow process. Claim 1 has been amended to clarify the present invention. Specifically, the nature of the service objects has been clarified. As discussed above, the service objects include functionality to parse the object model to determine the basic object types and functionality to perform functions on each of the basic object types. Gudmundson does not disclose a system which parses object models or provides functionality for basic object types. The Modifiers in Gudmundson provide no parsing function. It would perform no purpose in the system of Gudmundson. The Modifiers in Gudmundson are combined to provide characteristics to specific instantiations of Elements. They do not parse or act upon those Elements. Furthermore, Gudmundson does not provide a flow process as recited in claim 1. The flow process of the present claimed invention provides the steps of the application software through ordering of the selected service objects and data of the application. The data of the application is associated with the service objects for providing the functionality of the application through the flow process. Gudmundson does not disclose, teach or suggest a flow process relating an order for operation of service objects and application data. The only ordering in Gudmundson, as suggested in the Office Action, relates to the order of Modifiers within a specific instantiation of an Element. Finally, Gudmundson does not disclose, teach or suggest both object models and service objects which are finite in nature to accommodate infinite data structures. Accordingly, Gudmundson does not disclose, teach or suggest the present claimed invention. Claims 2-7 depend from claim 1 and are allowable for at least the same reasons.

Claim 8 recites a system for creating application software. The system includes means for storing a set of objection models, a stored set of service objects, means for selecting a subset of service objects, and means for defining a flow process. As discussed above, Gudmundson does not disclose, teach or suggest, object models, service objects or flow processes within the meaning of the present claimed invention. Accordingly, claim 8 patentably distinguishes over the cited art and is in condition for allowance.

Claims 9 and 10 depend from claim 8 and are allowable for at least the same reasons. Furthermore, claim 9 recites that the means for storing the object models includes: means for

receiving an application model, means for classifying each data element in the application model as a object model, and means for storing each object model. As noted above, complex data structures can be represented as sets of basic object types. Claim 9 recites a portion of the system which determines those sets from a representation of the application model. Gudmundson does not disclose, teach or suggest such a system. The Office Action references column 57, lines 1-5 as relating to this claim. However, that portion of Gudmundson merely recites that certain inherent object classes could be defined and operated outside of the core. That is unrelated to the invention as recited in claim 9. Nothing in Gudmundson receives an application model, classifies each data element in an application model as an object model, and stores object models. Therefore, claim 9 patentably distinguishes over the cited art.

Claim 11 recites a system for executing an application program. The system comprises a stored set of object models, an stored set of service objects, means for determining the basic object types of an object model upon execution of the service objects, and means for executing a function of each service object based upon the basic object types. As discussed above, Gudmundson does not disclose, teach or suggest, object models or service objects within the meaning of the present invention. Furthermore, Gudmundson does not disclose determining the basic object types of a data model and performing a function of a service object based upon the basic object types. Modifiers are associated with specific Elements. The core of Gudmundson processes the Modifiers to provide characteristics and behavior to the Elements. Gudmundson's system, however, does not process the Elements to determine the set of basic object types of the Element, or execute a function of a service object corresponding the each of the basic object types in the set. Accordingly, claim 11 patentably distinguishes over the cited art and is in condition for allowance.
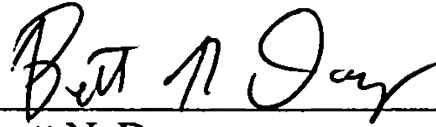
In view of the foregoing amendments and remarks, applicants believe that this application is in condition for allowance. If the examiner has any questions regarding this amendment or the application in general, he is encouraged to telephone the undersigned attorney so that prosecution of this application can be expedited.

Respectfully submitted,

Brett N. Dorny
Registration No. 35,860
Law Office of Brett N. Dorny
321 Church Street
Northborough, Massachusetts 01532
Telephone (508) 904-3228
Attorney for Applicant

Date: February 1, 2005

Correspondence should be addressed to customer number 43,891.